[RSSV19] B. Rumpe, I. Schaefer, B. Schlingloff, A. Vogelsang:
Special issue on engineering collaborative embedded systems.
In: SICS Software-Intensive Cyber-Physical Systems, 2019.
www.se-rwth.de/publications/

SPECIAL ISSUE PAPER

Special issue on engineering collaborative embedded systems

Bernhard Rumpe[1] · Ina Schaefer[2] · Bernd-Holger Schlingloff[3] · Andreas Vogelsang[4]

© Springer-Verlag GmbH Germany, part of Springer Nature 2019

1 Collaborative embedded systems (CrESt): the BMBF project

Welcome to the special issue on "Collaborative Embedded Systems" (CrESt). While in the old days software in embedded systems was isolated and mainly concentrating on predefined functionalities within the overall system context, software has now become the connecting factor between cooperating systems of systems and, because of its increasing functionality, also the driving complexity factor. Cyber-physical systems are composed of several autonomous individual systems, which have to collaborate in order to achieve common objectives. Collaboration of systems is necessarily based on collaboration of connected software components that control and govern the overall cyber-physical system behaviour.

In this context, the German ministry of research and education (BMBF) launched a large research initiative on development methods for collaborative embedded systems. The CrESt project (2017–2020) assembles more than twenty companies and research institutions, namely Bertrandt, Expleo, FEV, fortiss, Fraunhofer IESE and FOKUS, HSU Hamburg, HU Berlin, INCHRON, InSystems, itemis, Model Engineering Solutions, PikeTec, OFFIS, pure-systems, Robert Bosch, RWTH Aachen, Siemens, TU Berlin, TU Braunschweig, TU Kaiserslautern, TU München, and Univ. Duisburg-Essen. Their goal is to deepen the knowledge on how to develop collaborative embedded systems in the new challenging environments.

The CrESt project can be seen as a necessary extension of the projects SPES 2020 (2009–2012) and SPES_XT (2012–2015). These projects successfully developed concepts, methods and tools to master the challenges of today's complex embedded system development. The SPES 2020 methodology on model-based engineering of embedded systems, and its extension, the SPES_XT methodology, are documented in two Springer volumes and are now widely applied in industrial developments. The ambition of the CrESt project is to gain a holistic view on the additional challenges imposed by the fact that not only a single embedded system, but a whole group of collaborating systems is under development. On the one hand, it addresses foundational problems, e.g., how to formalize common goals and targets, what are appropriate modeling languages for collaborations, and which algorithmic solutions for monitoring the quality of the systems can be designed, to give early feedback to developers. On the other hand, it addresses practical challenges, such as open networks, system adaptation, open and dynamic system context, and how to design systems that shall be highly automated while still adaptive. These results are applied in various domains and on a number of examples that demonstrate the feasibility of the CrESt method, which substantially extends the previous SPES methods. This volume presents some of the foundational material and application examples which drive the development in CrESt.

2 Definitions

In order to demonstrate the CrESt concepts more precisely, we start with some definitions, which are used throughout all papers of the special issue.

An *embedded system* is a computational system which is a fixed part of a technical system. In this definition, a *technical system* is an artefact, which transforms or transmits matter, energy, and information, whereas a *computational system* is

✉ Bernhard Rumpe
rumpe@se-rwth.de
http://www.se-rwth.de

Ina Schaefer
schaefer@isf.cs.tu-bs.de

Bernd-Holger Schlingloff
hs@informatik.hu-berlin.de

Andreas Vogelsang
andreas.vogelsang@tu-berlin.de

1 RWTH Aachen University, Aachen, Germany

2 Technische Universität Braunschweig, Braunschweig, Germany

3 Humboldt-University and Fraunhofer FOKUS, Berlin, Germany

4 Technische Universität Berlin, Berlin, Germany

built to transform, transmit, and store *information* only. As an example for an embedded system, consider the control unit of an electric welding robot. The robot as a technical system uses energy to transform the filler wire into a welding seam. However, it could not do this without an embedded control system transforming information from the sensors into commands for the actuators, directing its movement and switching the welding power on and off.

Based on network availability as well as the need for automatic and efficient collaboration, technical systems are becoming more and more connected, forming complex *systems-of-systems*. In our example, the robot might not work on its own, but as a part of a production line, which is interconnected via a conveyor belt. Hence, also the embedded systems inside the production units have to be connected with one another. For instance, the robot has to communicate with the conveyor belt when the next item arrives. A *cyber-physical system* (CPS) is a set of technical systems in which their embedded computational subsystems are (intensively) interconnected. An embedded system is called *collaborative*, if it is part of a cyber-physical system where all components strive to accomplish a common task. In the example, all production units and conveyor belts in the factory collaborate to produce some goods. Hence, a ***collaborative embedded system*** (CES) is always part of a *collaborative system group* (CSG), which is a cyber-physical system with a common goal.

In order to cope with the complexity in the development of embedded systems, *model-based development* or *model-based systems engineering* (MBSysE) has been suggested. While stand-alone software can in principle be developed without explicit modelling, all other engineering disciplines participating in a systems development need to analyse the system properties on models of the system under development. Model-based development is therefore a natural and necessary approach to manage the complexity arising from CES. MBSysE is an engineering method, taking many ideas from model-based software engineering (MBSE), where a model of the system under design is used as a central representation, from which all other artefacts are derived. Here, a *model* is an *abstract description* of an existing or intended system, which serves a specific purpose. In MBSysE, the purpose of the model is to facilitate the design and implementation of the system to be built. Abstract models describe only those aspects and only with the necessary degree of detail which are essential for the design and understanding that the final system will meet the requirements, but the models still leave room for implementation choices. During the design and implementation process, these choices are gradually resolved, and the abstract *design model* is stepwise refined to a more concrete *implementation model*. If the implementation model is sufficiently concrete, executable code can be generated from it automatically.

Models are made explicit using dedicated *modelling languages*. Explicit models are necessary to communicate between developers, but also between developers and computers. Only then smart algorithms are able to identify deficits in the models early, by analyzing various properties or simulating the behavior of the intended cyber-physical system including communication delays, network errors, functional failures, etc. This way the ability of a cyber-physical system to collaborate actively in achieving common goals can be analysed far before any physical part of the system is built.

In computing science, many different modelling formalisms for software and systems have been designed and several of them have materialized as general purpose modelling languages, such as *UML* or *SysML*. Other formalisms have led to domain specific languages (*DSLs*) dedicated only for small and focused purpose. Some of these are referenced in this special issue. For the modelling of CES, it is important to be able to describe goals and intentions. Although there have been several attempts in goal modelling, so far none of them have been integrated into an overall continuous design methodology for CES. Furthermore, to manage adaptation and flexible reaction on changing contexts that collaborative embedded systems typically face, it is not only necessary to model goals and intentions during design, but also to transfer parts of these into runtime, while at the same time ensuring safety without compromises. The purpose of this special issue is to present some results towards such an integrated methodology.

3 Use cases in the CrESt project

For the derivation and presentation of results, throughout this special issue four different industrial use cases from the industrial CrESt partners are used. These are vehicle platooning, changeable factories, distributed energy production, and autonomous transport robots.

In the platooning use case, we consider a group of vehicles which share the goal to travel to a common destination. By driving in a low distance formation, the overall air resistance is decreased, and fuel consumption is significantly reduced. Furthermore, more vehicles fit onto the street and traffic may be more efficient. However, in order not to crash into one another, the vehicles constantly have to communicate. Scenarios within this use case are forming and dissolving the platoon, as well as joining and leaving the platoon by single vehicles.

The use case on changeable factories deals with flexible production cells, which collaborate to build products on demand. Each cell is able to build various components in different configurations, which are assembled according to changing customer needs. The common goal is to optimize

the use of production resources and machines for different usage scenarios.

In distributed energy production, various renewable power plants collaborate to guarantee a continuous supply of electric power to the customers. A challenge is to balance the overall energy production such that several side conditions are met. Collaboration here must be extremely quick.

Autonomous transport robots are driverless vehicles for loading and unloading production units in a factory. In a decentralized control scenario, each robot can decide which transport job to accept and accomplish. The common goal is to keep the production going, i.e., no machine may ever stop due to lack of supply material or abundance of processed material.

4 Results presented in this special issue

This special issue contains five articles describing research conducted as part of the CrESt project. They cover the broad spectrum of topics that is addressed in the project including *uncertainty handling*, *runtime verification*, *self-adaption*, *system simulation*, and *strategic collaboration*.

Constantin Hildebrandt, Torsten Bandyszak, Ana Petrovska, Nishanth Laxman, Emilia Cioroaica, and Sebastian Törsleff present an uncertainty classification scheme for information exchanged at runtime in their article "EURECA: Epistemic Uncertainty Classification Scheme for Runtime Information Exchange in Collaborative System Groups". The authors focus on epistemic uncertainty, which refers to the knowledge that is available to the system, for example, in the form of an ontology. They present a classification scheme that can be used to identify the relevant epistemic sources of uncertainties for a CES during requirements engineering.

In the second article "(Self)-Adaptiveness for Manufacturing Systems - Challenges and Approaches", Birte Böhm, Florian Grigoleit, and Stephan Unverdorben identify and elaborate challenges and approaches for self-adaptiveness in manufacturing systems. The main challenges are designing flexible production sites, representing manufacturing knowledge, as well as organization and implementation of (self-)adaptive manufacturing systems. Promising techniques to implement (self-) adaptiveness are system modularization, architectural patterns for flexible systems, and automatic reconfiguration of manufacturing systems.

Damian Kurpiewski and Diego Marmsoler investigate the use of strategic logics for the analysis of CESs in their article "Strategic Logics for Collaborative Embedded Systems - Specification and Verification of Collaborative Embedded Systems using Strategic Logics". They show that strategic model checking is useful to investigate certain aspects of CESs, such as the impact of environmental changes. They also show limitations of the approach, when it comes to the analysis of implementation-level aspects, such as performance.

Such low-level properties may be more interesting to study in simulations. Emilia Cioroaica, Florian Pudlitz, Ilias Gerostathopoulos, and Thomas Kuhn provide an overview of simulation methods and tools for design and runtime evaluation of groups formed by CES in their article "Simulation Methods for Collaborative Embedded Systems". They present solutions and challenges brought by evaluating vehicle collaboration using simulation, and suggest further directions of research and development in order to address gaps and to extend the applicability of simulation methods for collaborative embedded systems.

Finally, Samira Akili and Felix Lorenz address the challenge of runtime verification in CES in their article "Towards runtime verification of collaborative embedded systems". They present a case study based on industrial transport robots and model the main operating procedure, a distributed bidding protocol. The key properties that must hold for functional correctness turn out to comprise a large number of different semantic concepts that can not be jointly expressed with any single formalism. To address this issue, they identify three specification languages that are particularly suitable for monitoring of collaborative embedded systems: Certifying distributed algorithms, trace expressions, and real-valued temporal logic.

Acknowledgements First of all, the editors would like to thank the authors of the articles in this special issue as well as the reviewers who helped improving the quality of the articles. CrEST as well as its predecessor projects would not have been possible without a large number of people helping making it such a success. Foremost, we would like to thank Manfred Broy for the continuous steering of the projects towards a well integrated and successful methodology accompanied with appropriate tooling and applied in various industrial domains. Wolfgang Böhm has constantly ensured that progress is made both individually and towards the large overall goals. We thank Michael Weber (BMBF) and Dirk Günther (DLR Projektträger) for their support, the industrial partners for their participation both in applying SPES and CrESt methods in their industrial contexts, and all the scientific members of the academic partners for carrying out this research.